Add a new payment methode plugin to Bakery

This brief tutorial refers to Bakery v1.5.7 or later.
Tutorial v0.4

Files used in the plugin

A Bakery payment method plugin consists of at least 5 files and the language directory:

- **index.php**
- **info.php**: Contains payment method variables and info about the payment method
- **processor.php**: Prepares the data sent to the payment gateway api
- **report.php**: After successful payment transaction it receives a response from the payment gateway api in the background to set the payment status to 'paid'. Depending of the payment gateway this file can be named differently.
- **check_payment.php**: Checks whether the payment has been canceled by the user, the payment has been completed successfully or an error occurred during payment processing.
- **languages-directory**: Contains the localisation files of the payment method.
- **icon.php**: This icon will be used in the order administration of the Bakery backend to indicate the payment method that has been selected by the customer.
- **install.php / upgrade.php**: These files are optional and can be used to execute any code on payment gateway installation or upgrade.

Follow the steps below to implement a new payment gateway plugin:

1. Sign up to a test/sandbox account of your favorite payment service provider.

2. Read the developer's guide provided by the payment service provider. Especially read carefully about:
• How data is transfered to the payment gateway api (HTML, XML, HTTP, ...)
• How the payment gateway api confirms the successfully completed payment
• How the customer is redirected from the payment gateway back to the shop site

3. Check the code of the existing plugins provided in the payment_methods directory of Bakery. Choose an existing plugin that will fit best the requirements of the payment gateway you are going to implement.

4. Change the name of the directory to the name of your payment gateway.

5. info.php
First define the variables $field_1 to $field_6. Provide variable values like merchant id, merchant key, user id, partner id, project id, customer email or whatever your payment service provider requires for a valid authentication.

The shop admin will be asked to enter the required information in the Bakery backend.

Make sure you create for each value of the variable $field_X a corresponding variable in the payment method language files.

Example: $field_1 = 'email';
The value 'email' will be converted to uppercase 'TXT_EMAIL', which you can use in the language files array as follows:
$MOD_BAKERY[$payment_method]['TXT_EMAIL'] = 'E-Mail';

Second provide information about your payment method plugin using the variables
$payment_method_name = '';
$payment_method_version = '';
$payment_method_author = '';
$requires_bakery_module = '';

6. process.php
This file prepares and sends the required information to your payment gateway api as well as it contains a template that will be displayed as part of the checkout page. Customers get information about the available payment methods and can select the payment method of their choice.

Depending on the variety of data and the specified method how data will be transferred to the payment gateway api you have to modify this file. You might make use of a php class or any other sample code provided by your payment service provider to ease integration

If you need to handle payment method errors (in another file) redirect your customers back to the shop. Attach the var pay_error and an error number to the url using the get method:
header('location: ' . $_POST['setting_continue_url'] . '?pay_error=1');
Catch the error numbers in the process.php file and trigger the corresponding error messages:
$_GET['pay_error'] = 1;

```
$_GET['pay_error'] = 2;
```

7. report.php (File name can differ depending on the payment gateway)
This file is optional. It receives a response from the payment gateway api in the background using post/get or xml after the payment has been completed successfully. If so the database field payment_status will be set to 'paid'.

8. check_payment.php
This file checks whether the payment has been 'canceled' by user, the payment has been 'completed' successfully, the payment is 'pending' or whether an 'error' has occurred during payment processing. It sets the var $payment_status that will be used by the view_confirmation.php file to show the corresponding message on the confirmation page.

If your payment service provider supports the payment status 'pending', it sends an email to the customer and another to the shop merchant and displays a "Payment transaction will be processed shortly" message. The shop email will contain a notification on the pending payment. Do not forget to set the two language variables
$MOD_BAKERY[$payment_method]['TXT_PENDING']
and
$MOD_BAKERY[$payment_method]['TXT_TRANSACTION_STATUS']

In case of 'success' it concludes the order, sends emails to customer and merchant and displays the "Payment completed successful" message.

9. frontend.css
Add a new class at the end of the BUTTONS section. Eg:

```
.mod_bakery_bt_pay_newpaymentgateway_f {
	width: 98%;
}
```

Replace 'newpaymentgateway' by the directory name of your payment gateway.

10. icon.png
Provide a 16x16px png icon of your payment gateway.

11. Optional: Localized payment method name
If your payment method uses different names depending on localisations, add a new var with the payment method name to the corresponding Bakery language file. In this example I will use 'new':
$MOD_BAKERY['TXT_PAYMENT_METHOD_NEW'] = 'Neu';

Locate the comment below in the file /bakery/modify_payment_methods.php two times:
// If needed replace payment method names by localisations

Add a new line in the switch statement for your new payment method. Eg.
case 'new': $name = $MOD_BAKERY['TXT_PAYMENT_METHOD_NEW']; break;

12. Optional: install.php / upgrade.php
Use an install.php / upgrade.php file if your payment method needs eg. an additional field in the customer table or for whatever you need to do on installation or upgrade.